Evaluate Model Integrity Patterns

This document describes the boundaries between key subsystems in the domain model and the model-integrity patterns applied to maintain separation, clarity, and consistency.

1. Overview

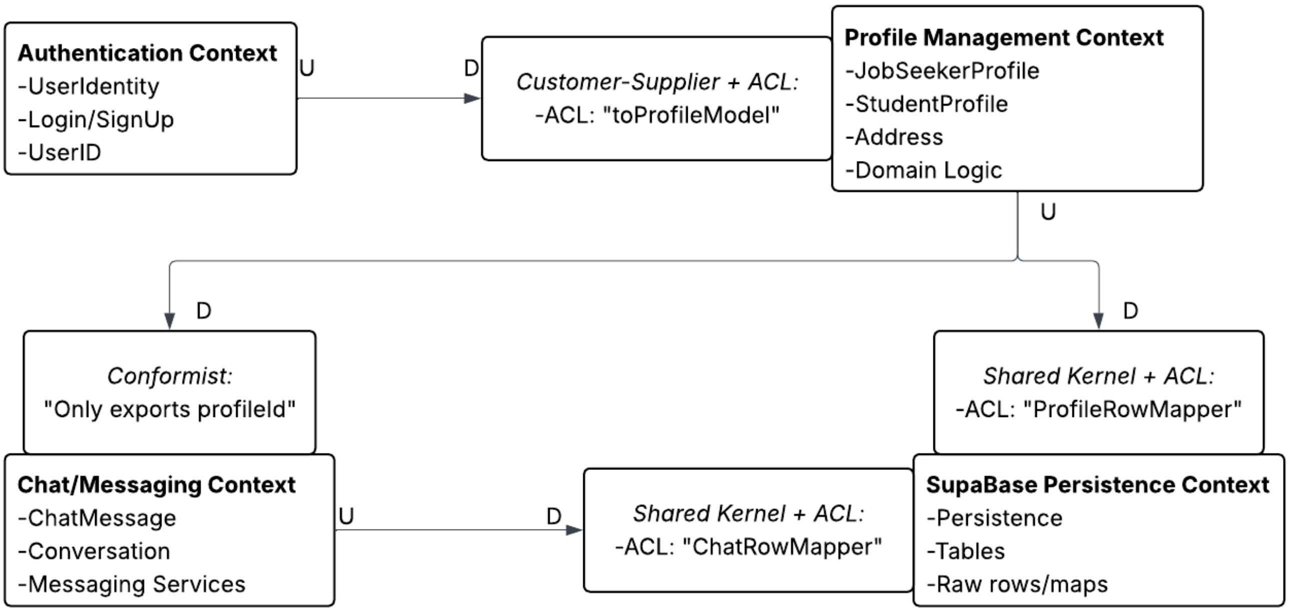
This document details:

- The subsystems (bounded contexts)
- The upstream/downstream relationships between them
- The model-integrity patterns applied (ACL, Shared Kernel, Customer–Supplier, Conformist)
- The rationale behind each pattern
- A diagram which is included below to visually represent the context map.

2. Context Map Diagram

https://lucid.app/lucidchart/29119dea-ad7e-4e32-a8de-109aa8e8b1e0/edit?viewport_loc=-160%2C-198%2C2265%2C1135%2C0_0&invitationId=inv_1a6ad0d4-b2c3-4cdc-afd2-25b23f3ea36b

or



3. Bounded Contexts and Patterns

3.1 Authentication Context

Responsibilities

- UserIdentity
- Login / SignUp
- UserID

Upstream

- None

Downstream

- Profile Management Context

Pattern Applied

- **Customer–Supplier + ACL**
  - ACL: `toProfileModel`
  - Authentication supplies stable identity data; Profile consumes it with translation.

3.2 Profile Management Context

Responsibilities

- JobSeekerProfile
- StudentProfile
- Address
- Profile domain logic

Upstream

- Authentication Context

Downstream

- Chat / Messaging Context
- Supabase Persistence Context

Patterns Applied

- **Conformist** (toward Chat/Messaging Context)

  - Chat accepts `profileId` and profile structure.

- **Shared Kernel + ACL** (toward Supabase Persistence Context)

  - ACL: `ProfileRowMapper`.
  - Shared domain-stable mapping rules.

3.3 Chat / Messaging Context

Responsibilities

- ChatMessage

- Conversation
- Messaging services

Upstream

- Profile Management Context

Downstream

- Supabase Persistence Context

Pattern Applied

- **Shared Kernel + ACL**
  - ACL: `ChatRowMapper`

3.4 Supabase Persistence Context

Responsibilities

- Persistence access
- Database tables
- Raw rows / maps

Upstream

- Profile Management Context
- Chat / Messaging Context

Downstream

- None

Pattern Applied

- **Shared Kernel**
  - Supports consistent persistence across contexts.

---

4. Why These Patterns?

- Avoid tight coupling between domain areas
- Create clear ownership of data and responsibilities
- Allow each context to evolve independently
- Provide controlled translation through ACLs
- Keep persistence logic separate and stable
- Ensure messaging and profile contexts stay clean and domain-focused
- Reduce cross-context assumptions
